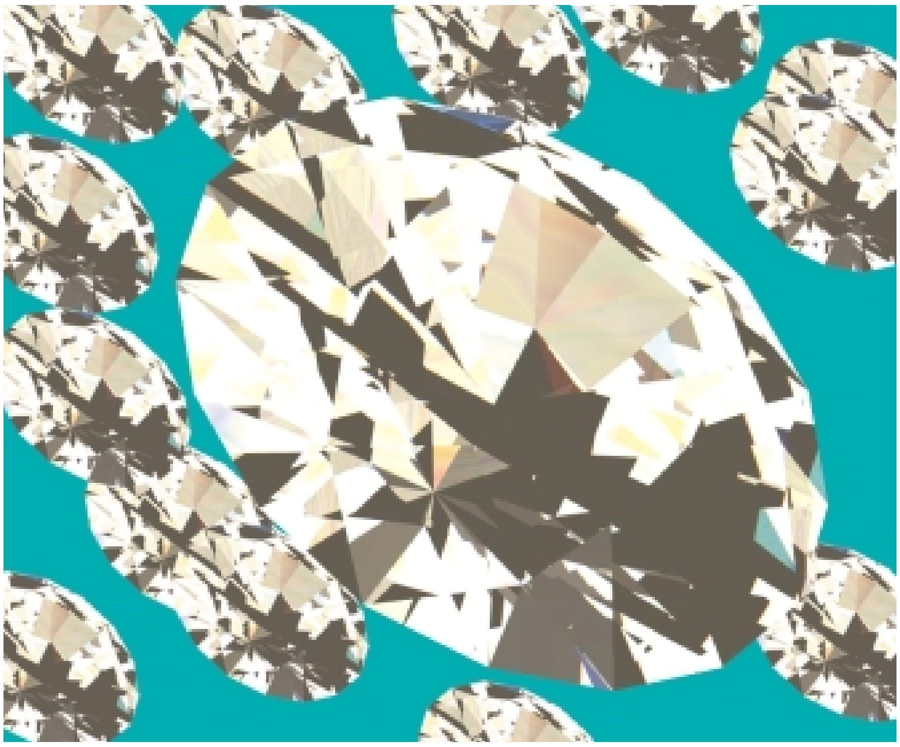
BWidgets: New GUI Elements for Tcl/Tk

Sparkling Desktop

BWidgets add common GUI elements such as tree and notebook to the standard Tk widgets. In contrast to most alternatives they were programmed entirely in Tcl/Tk and are thus well suited to platform independent applications. BY CARSTEN ZERBST



The Tcl GUI toolkit Tk comprises a complete selection of standard widgets, such as frames, buttons and listboxes. But some applications still need more. Additional widgets, such as tree or combobox are available in various add-ons, such as BLT[1], however, they tend to be precompiled, and thus platform specific.

BWidgets do not have this disadvantage as they were written in Tcl/Tk and can thus be distributed with the application across platform boundaries.

BWidgets are based on a development by Unifix and are now available from Sourceforge [2] under the BSD license. After extracting the archive you need to assign the add-on to the interpreter. To do so, you use the library path variable, “auto_path”:

```
lappend auto_path [file join ↵
//usr local lib BWidget-1.4.1]
```

The archive not only contains the add-on itself but also the comprehensive documentation, which you will find in the “BWMan” subdirectory. The interfaces are similar to those of the normal Tk widgets, however, the widget commands are capitalized.

Grouping and Organizing Widgets

Some BWidgets are used for organizing graphical desktops:

- Notebook (also referred to as a tab)
- Frame with frametext
- Vertically or horizontally tiled panes with modifiable partitioning
- Scroll areas

The Notebook widget (see Figures 1a and 1b, commonly referred to as a tabbed widget) allows the user to toggle between tabs. This in turn allows for more intuitive use. You can use the following syntax to add a tab:

```
set frame [$notebook insert ↵
end name -text "Example"]
```

The new frame can accomodate additional widgets. Tabs can of course be accessed via the mouse. The program can also specify the foreground tab:

```
$notebook raise name
```

Tabs can also be queried, sorted or deleted, but notebooks have limitations if you need to add a larger number of entries. Depending on the tab titles, there may not be enough room to display even four or five tabs simultaneously. The widget will then display buttons that allow you to click through the tabs (Figure 2), although this will affect the application’s handling.

You can solve this issue by using a page manager with a menu or a toolbar that allows the user to toggle the available tabs.

Strict division of desktop areas, as provided by notebooks, is not always necessary. A title frame, a frame with a title bar, as shown in Figure 1a is slightly less obtrusive.

However, Tk 8.4 supports the more powerful label frame, which is easier to use and allows you to assert more influence on its appearance. You can even use widgets in the title.

Divide and Conquer

Horizontally or vertically tiled areas are common GUI elements. The user can then use a button (sash) to determine the space assigned in each desktop area. The widget to use in this case is the paned window. The "-side" option allows you to specify the tiling direction and the position of the sash. Just like the "grid" command you use the "-weight" option to specify the tiling proportions. You can add an area to a paned window by typing "set frame [$panedWindow add]", and then place your own widgets in the new frame.

Automatic Scrollbars

When you move a paned window, at least one area is normally too small to accomodate all the widgets the window contains. But there is an elegant solution to this issue – that is, to use a scrolled window ("$scrolledWindow setwidget $child") that automatically displays scrollbars when required. However, only a few widgets, such as "text" or "canvas" will work with scrollbars. The scrollable frame defines a scrolling BWidget container, capable of accomodating any widget.

The BWidgets package traditionally duplicates some normal Tk widgets. We will not be looking into these widgets, as there is virtually no difference between their functionality and that provided by the native Tk widgets.

Needful Things

The additional input features not available in normal Tk, Combobox and Spinbox, are particularly interesting. A spinbox will become available in Tk 8.4, but this still leaves the combobox as an important input widget (see Listing 2). The widget allows the user to choose from a list containing presets defined by the "-values" option. The selection list can be uniquely predefined or – as shown in line 8 of our example – you can use a callback to update the list before displaying it.

If the user selects a new value, the widget calls a further callback function. The ".combo getvalue" function (line 12) will return the selected item, allowing you to access the element via "lindex". As line 25 of the sample program shows, the selected element can also be controlled via Tcl. In addition to the numeric position ".combo setvalue *@Position*", "last", "first", "next" or "previous" are also available. If the "-editable true" flag has been set, the user can also use the input field to type a

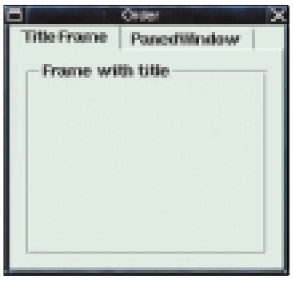


Figure 1a: Users can toggle between tabs in the notebook widget

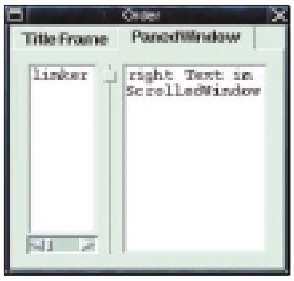


Figure 1b: The individual tabs can accomodate widgets, even paned windows

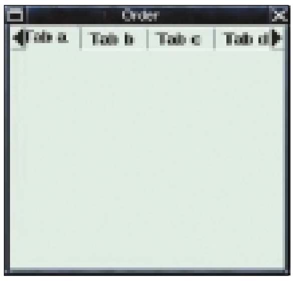


Figure 2: If there are too many tabs, two arrow buttons allow for scrolling

Listing 1: Base Widgets

```
#!/usr/local/bin/wish8.3
# Base Widgets from BWidgets

lappend auto_path [file join [pwd] BWidget-1.4.1]
package require BWidget 1.4.1

set notebook [NoteBook .nb]
pack $notebook -expand true -fill both

# First notebook with TitleFrame
set frame [$notebook insert end tf -text "TitleFrame"]
TitleFrame $frame.title -text "Frame with title"
pack $frame.title  -expand true -fill both

set f [$frame.title getframe]
label $f.label -text "an entry"
pack $f.label

# Second notebook with PanedWindow,
# ScrolledWindow and ScrolledFrame
set frame [$notebook insert end sp -text "PanedWindow" ]
set panedWindow [PanedWindow $frame.pw -side top ]
pack $frame.pw -expand true -fill both

set pane  [$panedWindow add  -weight 1]
set sw    [ScrolledWindow $pane.sw]
set text  [text $sw.text -wrap none -width 50 -heigh 50  -bg white]
$text insert 0.0 "right text in ScrolledWindow"
$sw setwidget $text
pack $sw -fill both -expand yes

set pane  [$panedWindow add -weight 9]
set sw    [ScrolledWindow $pane.sw]

pack $sw -fill both -expand yes
set sf [ScrollableFrame $sw.sf]
$sw setwidget $sf

set f [$sf getframe]
label $f.label -text "A label in a ScrollableFrame"
pack $f.label

# foreach t {a b c d e f g} {
#   $notebook insert end $t -text "Tab $t"
# }

$notebook raise [$notebook page 0]

wm title .  "Order"
wm geometry . 200x200
```

value directly. However, the callback function is not used in this case.

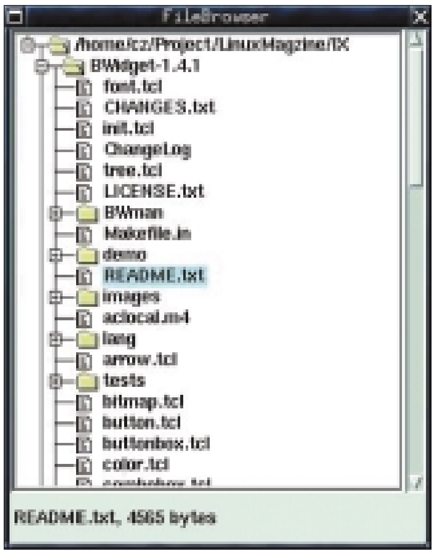
The drag & drop mechanism is a special case. The current Tk version does not contain a native implementation, however, you might prefer to use George Petasis' "tkdnd" [3] add-on, which supports native X11 and Windows drag & drop and is capable of communicating with Gnome or KDE applications.

Listing 2: Combobox

```
#!exec /usr/local/bin/wish8.3
# Example of BWidget Combobox

lappend auto_path [file join➥
[pwd] BWidget-1.4.1]
package require BWidget 1.4.1

proc selectionUpdate {} {
    .combo configure -values➥
[glob -nocomplain *]
}

proc valueChange {} {
    set index [.combo getvalue]
    set selection [.combo cget➥
-values]
    .value configure -text➥
[lindex $selection $index ]
}

ComboBox .combo -postcommand➥
selectionUpdate \
    -modifycmd valueChange➥
-editable false \
    -entrybg white

label .value
grid  .value .combo -sticky ew➥
-padx 10

selectionUpdate
.combo setvalue @0
valueChange

wm title .  "ComboBox"
```

Top of the Tree

Besides the structural elements, the tree display is one the most important features in BWidgets. Let's look at a simple file browser in Listing 3 as an example (see also Figure 3). Line 10 creates the widget and specifies both the colors and a callback function that is called on opening and closing a node. Trees can become extremely large, and this is why line 7 again reverts to the scrolled window. The first step is to



```
/home/cz/Project/LinuxMagazine/IX
  BWidget-1.4.1
    font.tcl
    CHANGES.txt
    init.tcl
    ChangeLog
    tree.tcl
    LICENSE.txt
    BWman
    Makefile.in
    demo
    README.txt
    images
    aclocal.m4
    lang
    arrow.tcl
    tests
    bitmap.tcl
    button.tcl
    buttonbox.tcl
    color.tcl
README.txt, 4565 bytes
```

Figure 3: This file browser was implemented using the BWidgets tree and shows the contents of a directory when you open the directory

insert a node into the tree; this step is performed using the command:

```
$tree insert index parent name
```

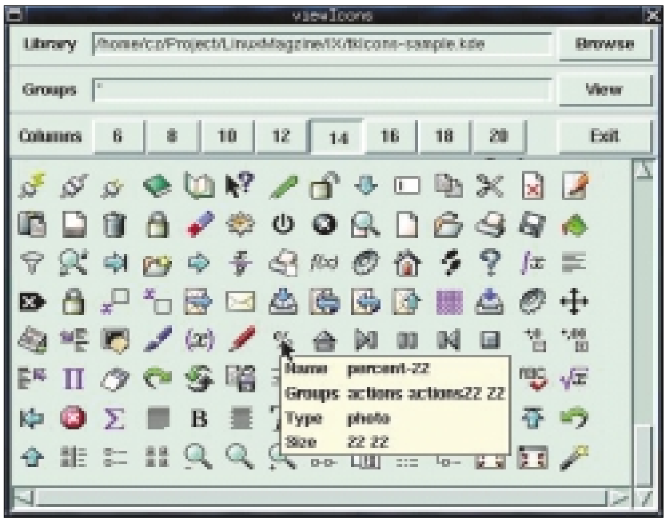
The index accepts the same values as the "lindex" command, mainly "end". The parent field designates the node under which the new node will be inserted – this will be "root" for the first node. The node name can be any string, however, the name must be unique amongst the tree widgets.

The node name is usually derived from the data to be displayed, although a serial number can be used. There are a few options available for nodes in addition to the text and image to be displayed, particularly "-data *value*", which can store any additional data in the node.

Current Content Only

The callback function "nodeOpen" (line 27) first deletes all the child nodes and then recreates them when a node is opened – thus ensuring that the file browser will always display the current content of the directory. If a user wants to use the mouse to open a subtree, she will need to click the small box next to the node. The box is either drawn automatically (as in the case of the first node) or explicitly using the "-drawcross always|never|auto" option.

Most applications will require the user to select a node. You can use "$tree bindText *Event Callback*" or "bindImage" to bind a callback function to an event



```
Library  /home/cz/Project/LinuxMagazine/IX/tkicons-sample.kde
Groups   *
Name    percent-22
Groups  actions actions22 22
Type    photo
Size    22 22
```

Figure 4: The Icon package by Adrian Davis not only contains a large number of icons, but also a browser that you can use to view the icon pool

Breaking News

On September 5 Tcl/Tk finally went to version 8.4. The sources for the beta version are available from [4], and pre-compiled packages for Linux from Activestate [5]. We will be looking at the new version's features in our next issue.

You can download the presentations from this year's European Tcl/Tk meeting, which was held in Munich, from Michael Haschek's web site. A variety of topics were discussed, ranging from e-learning to 3D graphics. The latter also looked into a new Tcl add-on by General Motors [7] that supports both tensors and the production of 3D graphics. By the way, Tcl/Tk users in Munich are currently founding their own user group.

The Bwdigets comprise a number of icons that can be queried using the (undocumented) bitmap command. For improved ease of use Adrian Davis has compiled a number of freeware icons – from KDE for example – in his "icon" package [8]. He also provides a browser for this purpose (see Figure 4)."

Listing 3: Tree Widget

```
#!/usr/local/bin/wish8.3
# The BWidgets tree widget

lappend auto_path [file join [pwd] BWidget-1.4.1]
package require BWidget 1.4.1

set sw [ScrolledWindow .sw -relief sunken➘
-borderwidth 2]
grid  $sw -sticky nesw

set tree [Tree $sw.tree -background white \
    -selectbackground LightSkyBlue \
    -opencmd nodeOpen \
    -closecmd nodeClose
]
$sw setwidget $tree
$tree bindText <Button-1> selected
$tree bindImage <Button-1> selected

label .label -textvariable fileInfo -anchor w
grid .label -sticky ew

grid columnconfigure . 0 -weight 10
grid rowconfigure . 0 -weight 10
grid rowconfigure . 1 -weight 1
# Callbacks
proc nodeOpen {node} {
    # Swap icon
    $::tree itemconfigure $node -image [Bitmap::➘
get openfold]
    # delete old child nodes
    $::tree delete [$::tree nodes $node]

    # Directory of nodes
    set path [$::tree itemcget $node -data]

    # Create nodes for all children
    foreach child [glob -nocomplain [file join➘
$path *]] {
        if {[file isfile $child]} {
            set icon [Bitmap::get file]
            set dc never
        } else {
            set icon [Bitmap::get folder]
            set dc allways
        }
        $::tree insert end $node $child -data➘
$child \
            -text [file tail $child] \
           -image $icon -drawcross $dc
    }
}

proc nodeClose {node} {
    $::tree itemconfigure $node \
        -image [Bitmap::get folder]
}

proc selected {node} {
    $::tree selection set $node
    set path [$::tree itemcget  $node -data]
    set ::fileInfo "[file tail $path], [file➘
size $path] bytes"
}

# insert first node
$tree insert end root pwd -data [pwd] -text [pwd] \
    -image [Bitmap::get folder]

# ... and open
$tree opentree pwd false

wm title . "FileBrowser"
```

that occurs for the text or the node icon.

The callback function is responsible for raising the selected element. The "$tree selection *subcommand*" with the subcommands "set", "get", "clear", "add" and "remove" takes care of this. The "selected" procedure (line 56) points the selection to a specific node and displays the name and size of the file by reference to the "::fileInfo" variable (the variable is displayed in the status line, see line 19).

Good Reasons

In addition to the features already mentioned, BWidgets also contains a progress indicator, as well as password, font and color dialog boxes. Additionally, you can use "DynamicHelp" to define help texts for menus. "Dialog" can be used as a template for dialog boxes of your own.

Although some add-ons are already obsolete there are several good reasons for using BWidgets: It allows easy programming of modern GUIs which you would be hard put to achieve working only with native Tk elements. ■

INFO

[1] BLT: *http://incrtcl.sourceforge.net/blt/*

[2] BWidgets: *http://tcllib.sourceforge.net*

[3] Tkdnd: *http://www.iit.demokritos.gr/~petasis/*

[4] Tcl: *http://www.tcl.tk/software/tcltk/8.4.html*

[5] Active Tcl: *http://aspn.activestate.com/ASPN/Downloads/ActiveTcl/*

[6] Presentations: *http://www.t-ide.com/tcl2002e.html*

[7] TK3D: *http://www.gm.com/automotive/innovations/rnd/TK3/TK3D_Software_Description.html*

[8] Icon: *http://www.satisoft.com/tcltk/icons/*

THE AUTHOR

Carsten Zerbst works for Atlantec on the PDM ship building system. He is also interested in Tcl/Tk usage and applications.